Davie Truong

CMPS 115

dtruong8@ucsc.edu

RAC2-Scrum and Engineering Practices: Experiences of Three Microsoft Teams

In the article “Scrum + Engineering Practices: Experiences of Three Microsoft Teams” published by *ESEM ’11 Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement,* on September 22 – 23, 2011, Laurine Williams from North Caroline State University and Gabe Brown, Adam Meltzer, and Nachiappan Nagappan from the Microsoft Corporation, discuss an agile software development process and how it can be improved. The authors begin by introducing the scrum methodology, informing the reader on its presence in the software engineering industry and how with lack of prescribed engineering practices, the methodology can lead to a “Flaccid Scrum”. “Coined by Martin Fowler, Flaccid Scrum is used to refer to teams that utilize only Scrum’s project management practices.” Flaccid Scrum teams often fail to pay enough attention to the quality of the code produced, creating the easiest scenario feature that can be considered done, thus resulting in mediocre code. To prevent flaccid scrum and inadequate project management, the developers from the Microsoft Corporation detail nine engineering practices in which Williams summaries.

Starting with a foundation, Williams describes the scrum process which can be simplified into: creating features, coding in iterations called sprints, holding sprint meetings to highlight required features, daily scrum meetings to prevent overlap, sprint reviews to demonstrate complete features, and a retrospective meeting to improve on the next sprint. With the understanding of scrum, Williams describes the engineering practices and discusses each with detail, revealing how each can improve quality or productivity. Planning poker is utilized to stream line the project tasks. The practice provides: a shared understanding; exposes hidden assumptions of technical aspects; a discussion for the implication throughout the system; surfacing and resolving ambiguities; and exposing easy and hard alternative for achieving desired goals. A benefit to Planning Poker voting is a deadlock, in which no consensus can be reach. This signals early in the sprint plan meeting that the product owner has not fully described the work to be thus further research must be done before taking the feature off the backlog. Continuous Integration is where members of a team integrate their work into a main build system frequently. The integration keeps a constant focus on quality, because builds and test runs are not always successful causing engineers to need to deal with issues such as bad merges, build system problems, and source control integration problems. Unit Test Driven Development is the process in which software engineer cycles on a minute by minute basis between writing failing automated unit tests and writing implementation code to pass those tests. Quality Gates, pre-established by the team, provide concise and measurable exit criteria for feature development putting the focus on quality of features rather than quantity of features. This practice is a key element in suppressing flaccid scum because it holds the coded features to a higher standard before it can be marked as complete. Source control is management of changes to documents, programs, and other information stored as computer files through a source control system. Code coverage required engineers to manage their automated unit test coverage and monitor the coverage with each build. Peer review allow teams to conduct design reviews of architecture diagrams and of code when adding new features. This helps improve the quality of code by removing faults that may have escaped. Static analysis tools can identify common coding problems or unusual code early in the development process. XML documentation keeps the code self-documenting. With the nine engineering practices detailed, Williams, concludes her article and case study with results.

Because of the scrum process paired with the nine engineering practices, two of the three teams that were in the case study that implemented the method had a lower defect density meaning a higher quality of code. Williams uses the defect density and one of the team’s productivity improvement of 250% to base her conclusion that the combination of scrum and engineering practices leads to improved productivity and product quality. From the article, the most crucial point was that the scrum methodology is not a perfect software development process and that quality gates ensure proper code is created during the development process.